计算机组成原理

第七章 指令系统

阙夏
计算机与信息学院
2021/5/26

大　纲

四． 指令系统

(一) 指令格式

1、指令的基本格式

2、定长操作码指令格式

3、扩展操作码指令格式

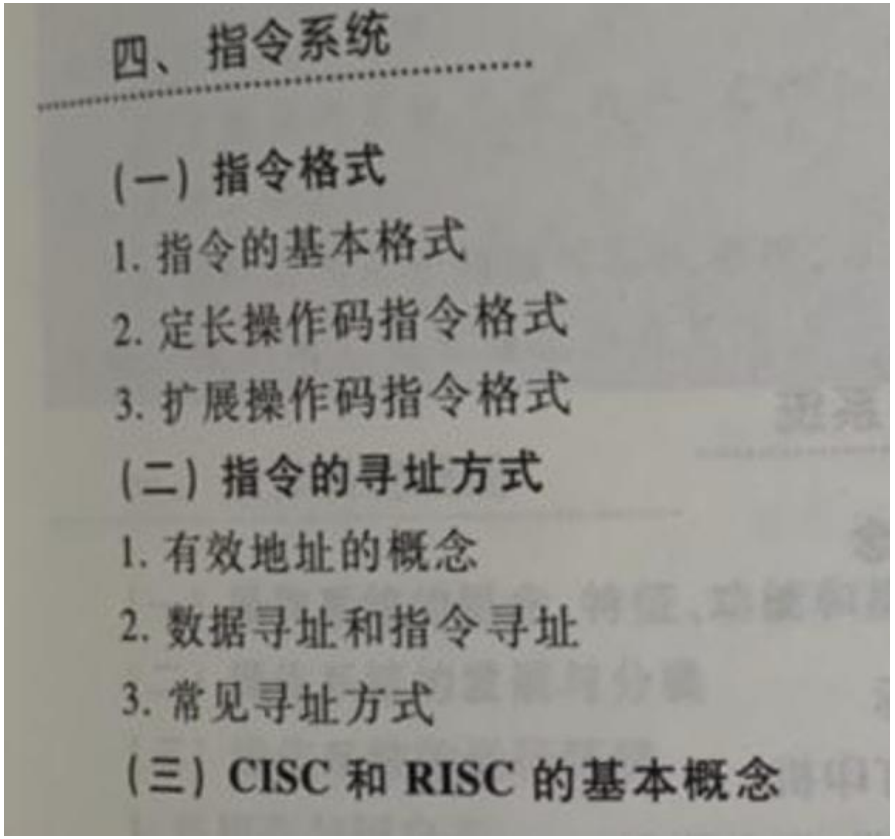
(二) 指令的寻址方式

1、有效地址的概念

2、数据寻址和指令寻址

3、常见寻址方式

(三) CISC和RISC的基本概念


四、指令系统

(一) 指令格式

1. 指令的基本格式

2. 定长操作码指令格式

3. 扩展操作码指令格式

(二) 指令的寻址方式

1. 有效地址的概念

2. 数据寻址和指令寻址

3. 常见寻址方式

(三) CISC 和 RISC 的基本概念

Contents

7.1 机器指令

一、指令格式

指令：就是让计算机识别并执行某种操作的命令。

指令系统：一台计算机中所有机器指令的集合。
称为这台计算机的指令系统。

系列计算机：指基本指令系统相同且基本体系，结构相同的一系列计算机。

系列机能解决软件兼容问题的必要条件是该系列的各种机种有共同的指令集，而且新推出的机种的指令系统一定包含旧机种的所有指令，因此在旧机种上运行的各种软件可以不加任何修改地在新机种上运行。

7.1 机器指令

指令的一般格式：

n位操作码字段的指令系统最多能够表示2^n条指令。

操作码字段	地址码字段

1. 操作码　指令应该执行什么性质的操作和具有何种功能。反映机器做什么操作。

(1) **长度固定**（译码简单）

用于指令字长较长的情况，RISC

如 IBM 370　　操作码 8 位

(2) **长度可变**

操作码分散在指令字的不同字段中

(3) 扩展操作码技术（教材P301）

操作码的位数随地址数的减少而增加

OP	A_1	A_2	A_3	
4 位操作码	0000	A_1	A_2	A_3	最多15条三地址指令
	0001	A_1	A_2	A_3	
	⋮	⋮	⋮	⋮	
	1110	A_1	A_2	A_3	
8 位操作码	1111	0000	A_2	A_3	最多15条二地址指令
	1111	0001	A_2	A_3	
	⋮	⋮	⋮	⋮	
	1111	1110	A_2	A_3	
12 位操作码	1111	1111	0000	A_3	最多15条一地址指令
	1111	1111	0001	A_3	
	⋮	⋮	⋮	⋮	
	1111	1111	1110	A_3	
16 位操作码	1111	1111	1111	0000	16条零地址指令
	1111	1111	1111	0001	
	⋮	⋮	⋮	⋮	
	1111	1111	1111	1111	

(3)扩展操作码技术

操作码的位数随地址数的减少而增加

OP	A_1	A_2	A_3

4 位操作码

0000	A_1	A_2	A_3
0001	A_1	A_2	A_3
⋮	⋮	⋮	⋮
1110	A_1	A_2	A_3

8 位操作码

1111	0000	A_2	A_3
1111	0001	A_2	A_3
⋮	⋮	⋮	⋮
1111	1110	A_2	A_3

12 位操作码

1111	1111	0000	A_3
1111	1111	0001	A_3
⋮	⋮	⋮	⋮
1111	1111	1110	A_3

16 位操作码

1111	1111	1111	0000
1111	1111	1111	0001
⋮	⋮	⋮	⋮
1111	1111	1111	1111

两个原则：
1.编码不能重复
2.短码不能为长码前缀

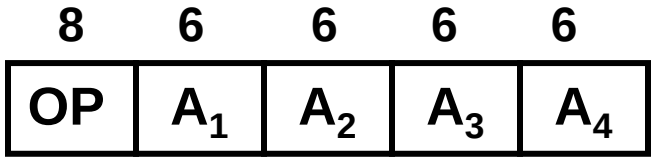
三地址指令操作码
每减少一种最多可多构成
2^4 种二地址指令

二地址指令操作码
每减少一种最多可多构成
2^4 种一地址指令

使用频率高用短码
使用频率低用长码

2. 地址码

(1) 四地址



A_1 第一操作数地址

A_2 第二操作数地址

A_3 结果的地址

A_4 下一条指令地址

(A_1) OP $(A_2) \longrightarrow A_3$

设指令字长为 32 位

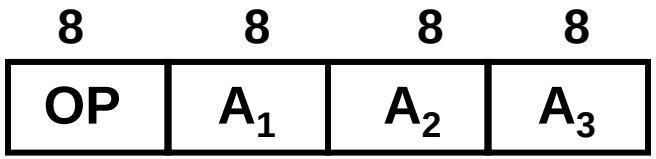
操作码固定为 8 位

4 次访存

寻址范围 $2^6 = 64$

若 PC 代替 A_4

(2) 三地址



(A_1) OP $(A_2) \longrightarrow A_3$

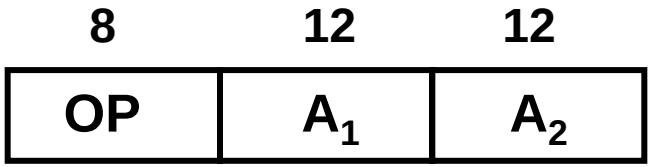
4 次访存

寻址范围 $2^8 = 256$

若 A_3 用 A_1 或 A_2 代替

7.1 机器指令

(3) 二地址



(A_1) OP $(A_2) \longrightarrow A_1$　　4 次访存

或

(A_1) OP $(A_2) \longrightarrow A_2$　　寻址范围 $2^{12} = 4$ K

若结果存于 ACC　　3次访存　　若ACC 代替 A_1（或A_2）

(4) 一地址

2 次访存

(ACC) OP $(A_1) \longrightarrow$ ACC　　寻址范围 $2^{24} = 16$ M

(5) 零地址　　无地址码

1.无需任何操作数，如空操作指令、停机指令等；2.所需操作数地址是默认的。

二、指令字长

指令字长决定于
- 操作码的长度
- 操作数地址的长度
- 操作数地址的个数

1. 指令字长 固定

指令字长 = 存储字长 = 机器字长（早期计算机）

2. 指令字长 可变

按字节的倍数变化

小结

- **当用一些硬件资源代替指令字中的地址码字段后**
  - 可扩大指令的寻址范围
  - 可缩短指令字长
  - 可减少访存次数
- **当指令的地址字段为寄存器时**

  三地址　OP　R_1，R_2，R_3

  二地址　OP　R_1，R_2

  一地址　OP　R_1

  - 可缩短指令字长
  - 指令执行阶段不访存

7.2 操作数类型和操作种类

一、操作数类型

地址	绝对地址：无符号整数；相对地址：有符号数
数字	定点数、浮点数、十进制数
字符	ASCII
逻辑数	逻辑运算

二、数据在存储器中的存放方式

例7.1 12345678H，分别用大端小端方式存储

存储器中的数据存放：边界对齐

字节地址

字地址				
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11

字节地址

字地址				
0	12H	34H	56H	78H
4	4	5	6	7
8	8	9	10	11

高位字节 地址为字地址

字节地址

字地址				
0	78H	56H	34H	12H
4	4	5	6	7
8	8	9	10	11

低位字节 地址为字地址

7.2 操作数类型和操作种类

三、操作类型

1. 数据传送

源	寄存器	寄存器	存储器	存储器
目的	寄存器	存储器	寄存器	存储器
例如	MOVE	STORE MOVE PUSH	LOAD MOVE POP	MOVE

置“1”，清“0”

2. 算术逻辑操作

加、减、乘、除、增 1、减 1、求补、浮点运算、十进制运算

与、或、非、异或、位操作、位测试、位清除、位求反

如 8086

ADD SUB MUL DIV INC DEC CMP NEG

AAA AAS AAM AAD

AND OR NOT XOR TEST

7.2 操作数类型和操作种类

3. 移位操作

算术移位　　逻辑移位

循环移位（带进位和不带进位）

4. 转移

(1) 无条件转移　JMP

(2) 条件转移

结果为零转　(Z = 1)　JZ

结果溢出转　(O = 1)　JO

结果有进位转　(C = 1)　JC

跳过一条指令　SKP

如

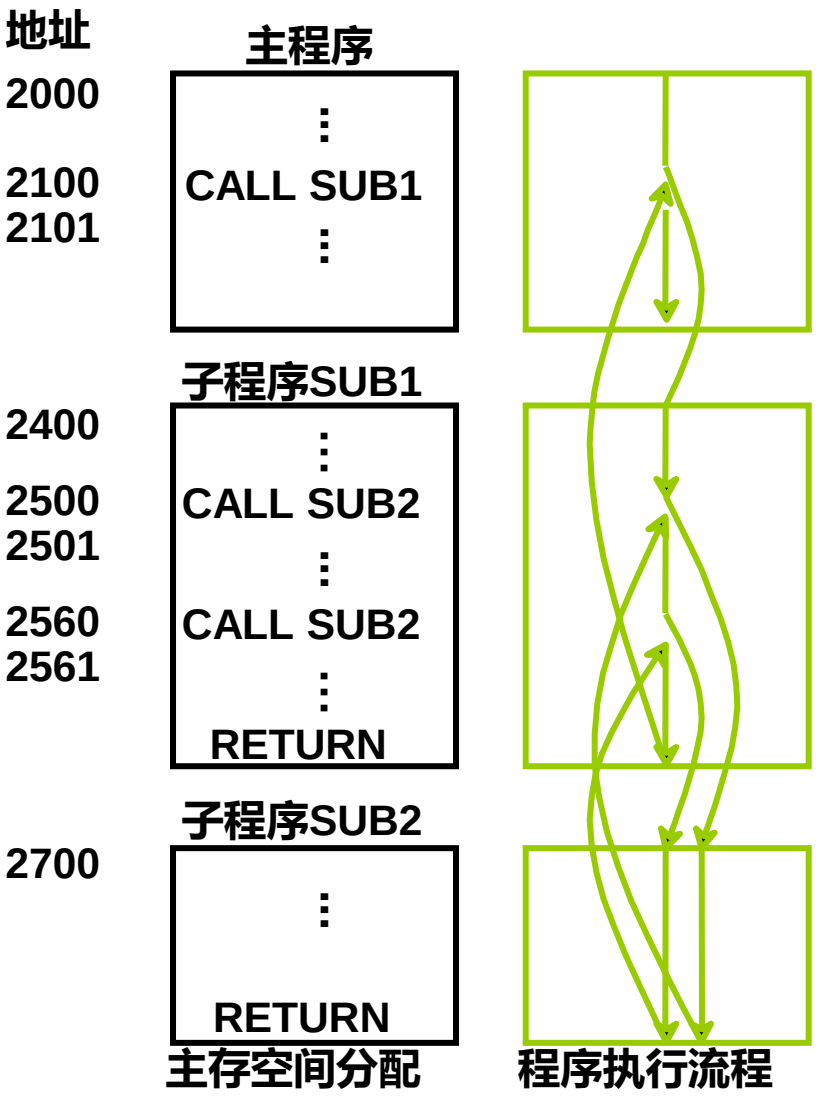
```
300
 ⋮
305
306
307
```

完成触发器

SKP DZ D = 0 则跳

7.2 操作数类型和操作种类

(3) 调用和返回



7.2 操作数类型和操作种类

(4) 陷阱（Trap）与陷阱指令

陷阱：计算机系统的意外事故（如电压不稳。故障等）。
陷阱指令：是处理陷阱的指令。

- 一般不提供给用户直接使用

  在出现事故时，由 CPU 自动产生并执行（隐指令）

- 设置供用户使用的陷阱指令

  如 8086　　INT　TYPE　　软中断

  提供给用户使用的陷阱指令，完成系统调用

5. 输入输出

入　　端口地址 ⟶ CPU 的寄存器

如　　IN　AL, *m*　　IN　AL, DX

出　　CPU 的寄存器 ⟶ 端口地址

如　　OUT　*n*, AL　　OUT　DX, AL

7.3 寻址方式

寻址方式

确定 本条指令 的 **操作数地址**

下一条 欲执行 指令 的 **指令地址**

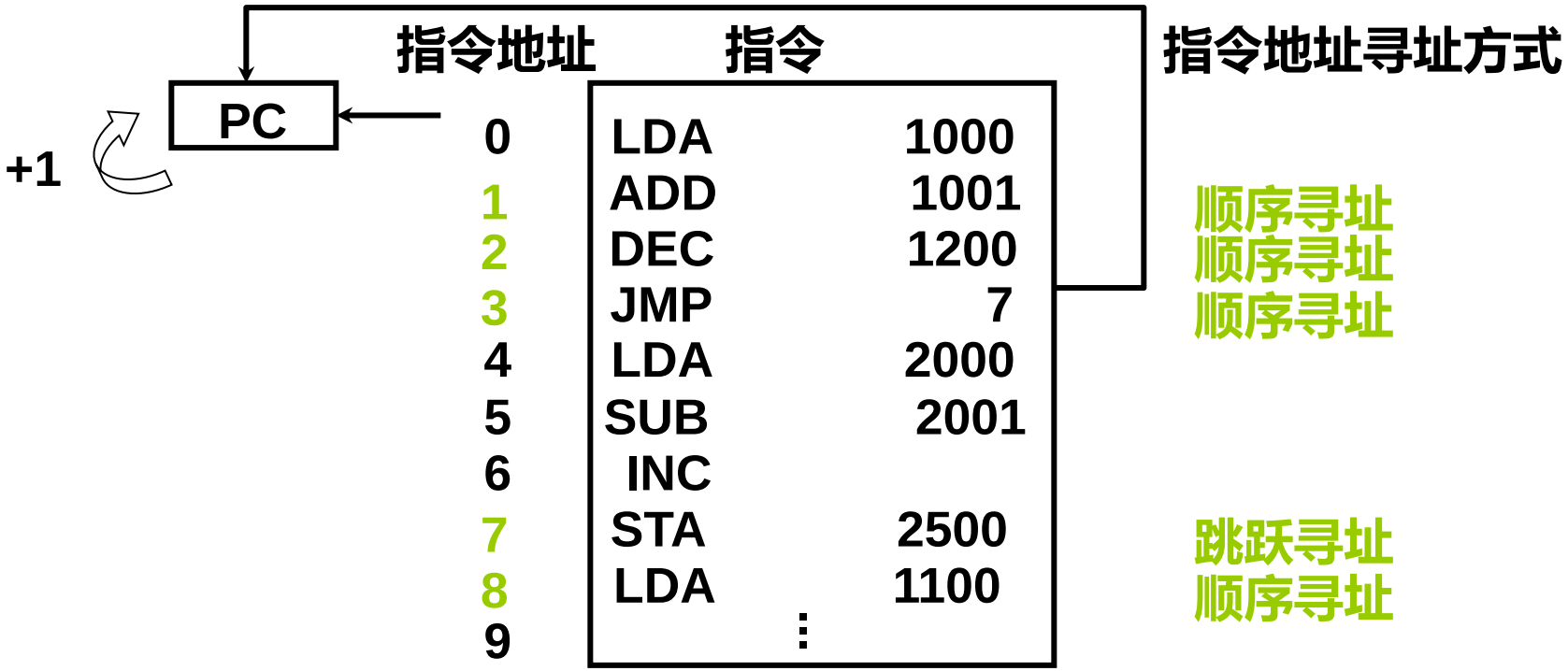
寻址方式
- 指令寻址
- 数据寻址

7.3 寻址方式

一、指令寻址

这里的+1是指下一条指令

- 顺序　(PC) + 1 ⟶ PC
- 跳跃　由转移指令指出，如 JMP



7.3 寻址方式

二、数据寻址

操作码	寻址特征	形式地址 A

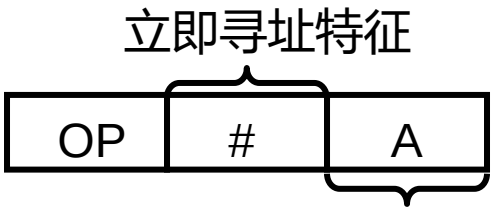
形式地址 指令字中的地址 **A**

有效地址 操作数的真实地址 **EA**

约定 **指令字长 = 存储字长 = 机器字长**

1. 立即寻址

形式地址 A 就是操作数



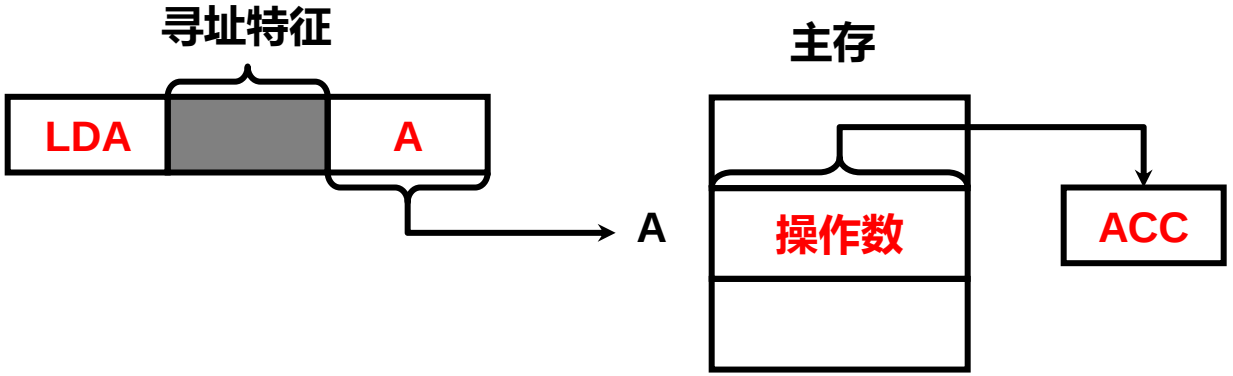
可正可负 **补码**

- **指令执行阶段不访存**
- **A 的位数限制了立即数的范围**

7.3 寻址方式

2. 直接寻址

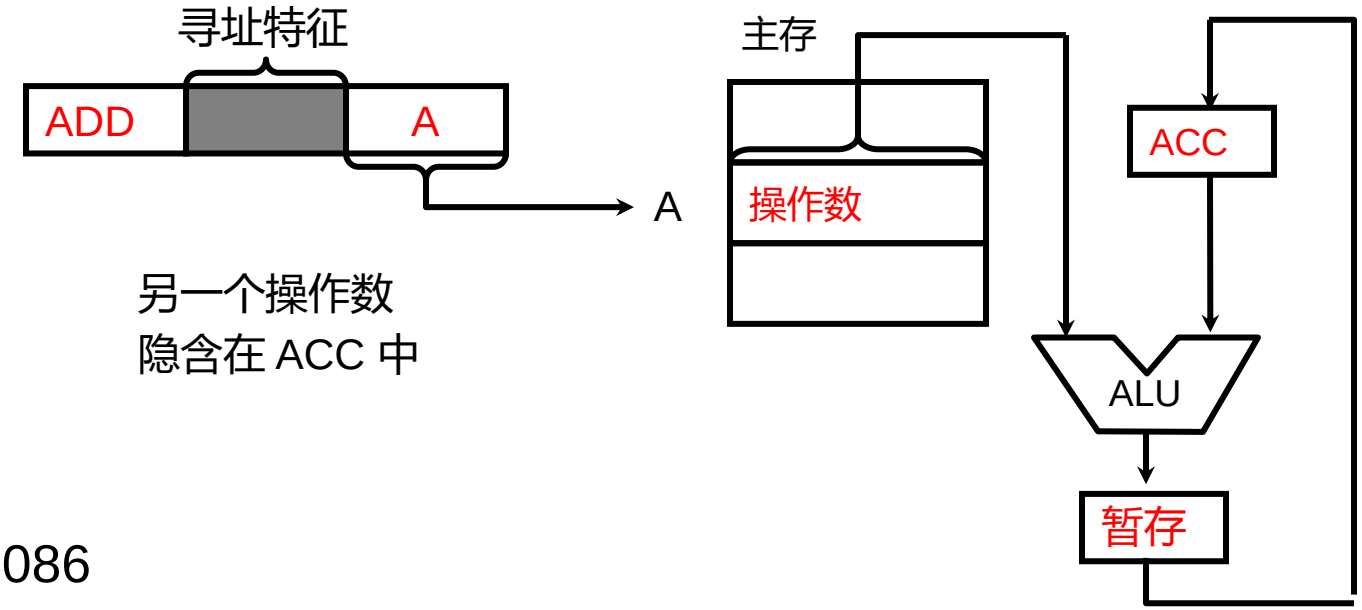
EA = A　　有效地址由形式地址直接给出



- 执行阶段访问一次存储器
- A 的位数决定了该指令操作数的寻址范围
- 操作数的地址不易修改（必须修改A）

3. 隐含寻址

操作数地址隐含在操作码中



如 8086

MUL 指令　　被乘数隐含在 AX（16位）或 AL（8位）中

MOVS 指令　　源操作数的地址隐含在 SI 中

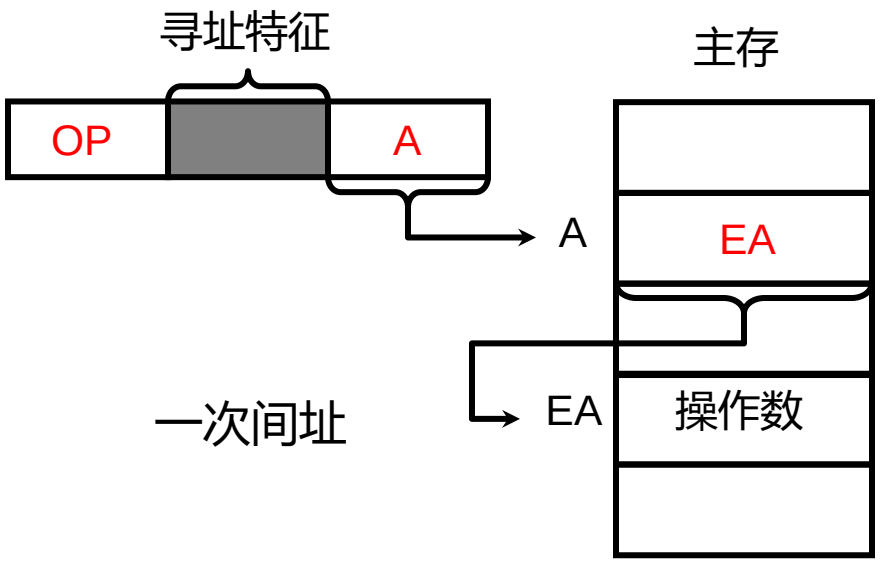
目的操作数的地址隐含在 DI 中

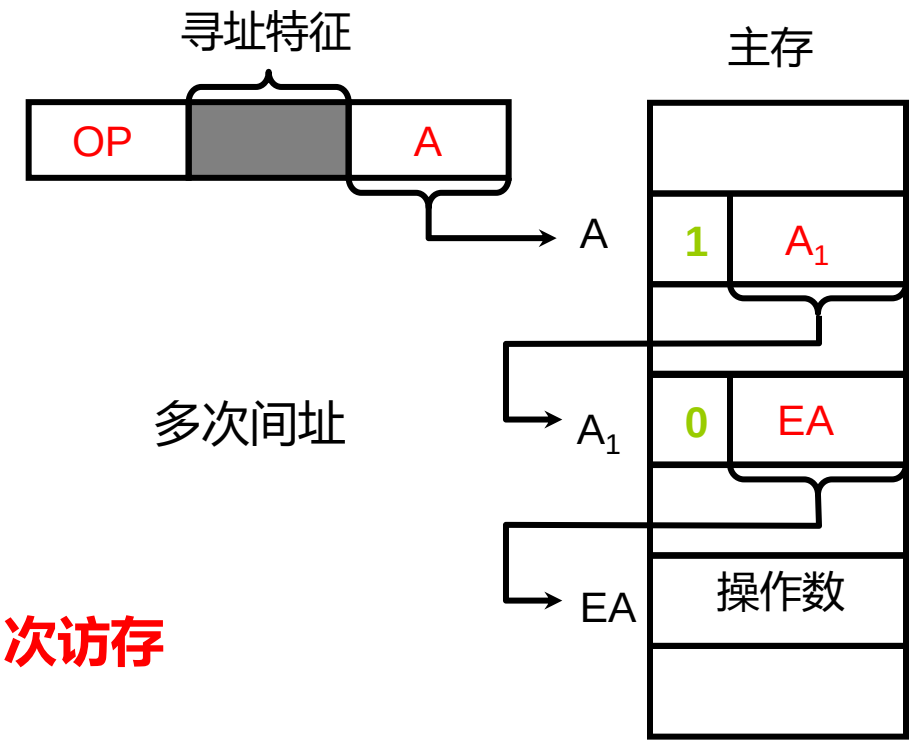
- **指令字中少了一个地址字段，可缩短指令字长**

7.3 寻址方式

4. 间接寻址

EA = （A）　有效地址由形式地址间接提供



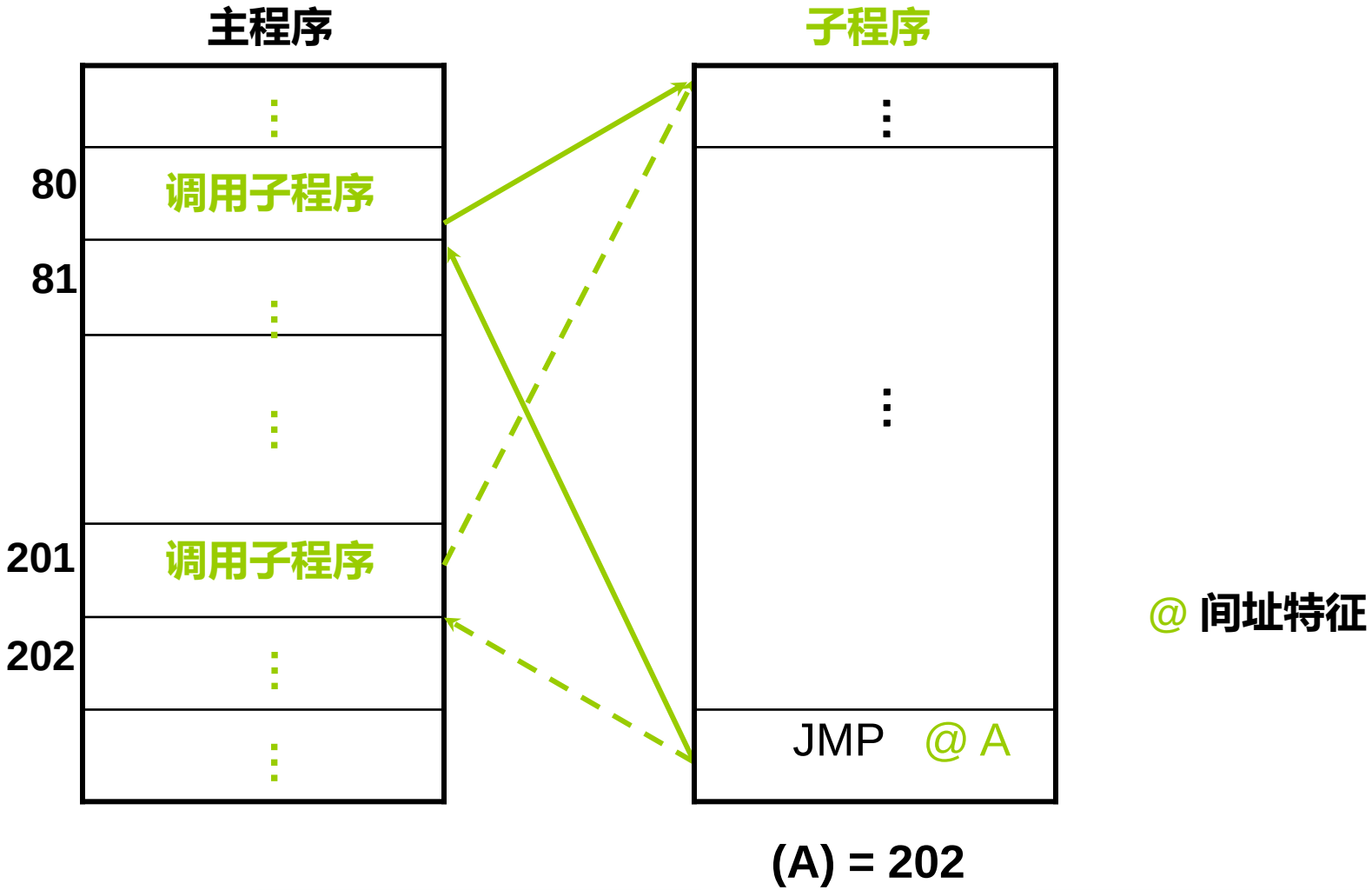


- 执行指令阶段 2 次访存
- 可扩大寻址范围
- 便于编制程序

多次访存

7.3 寻址方式

间接寻址编程举例

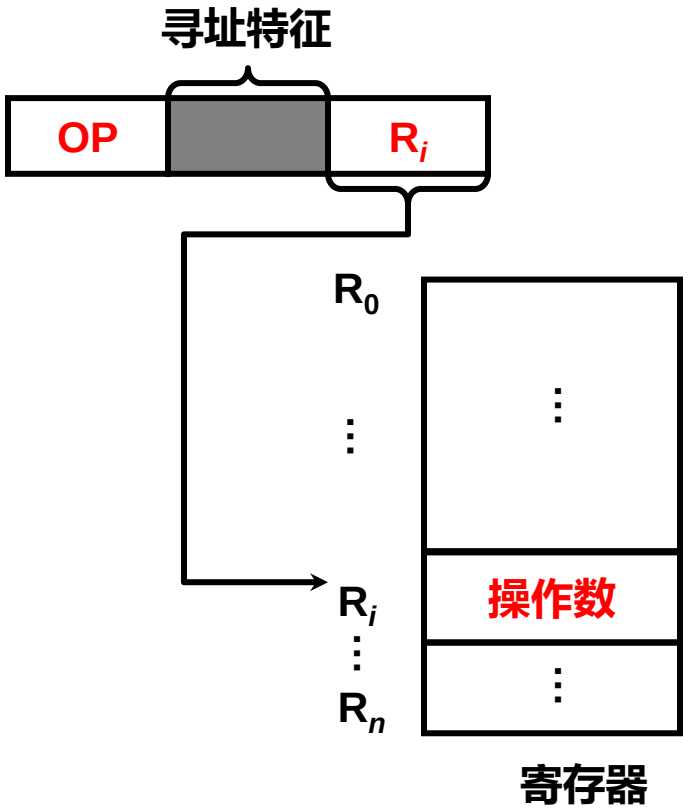


7.3 寻址方式

5. 寄存器（直接）寻址

$EA = R_i$

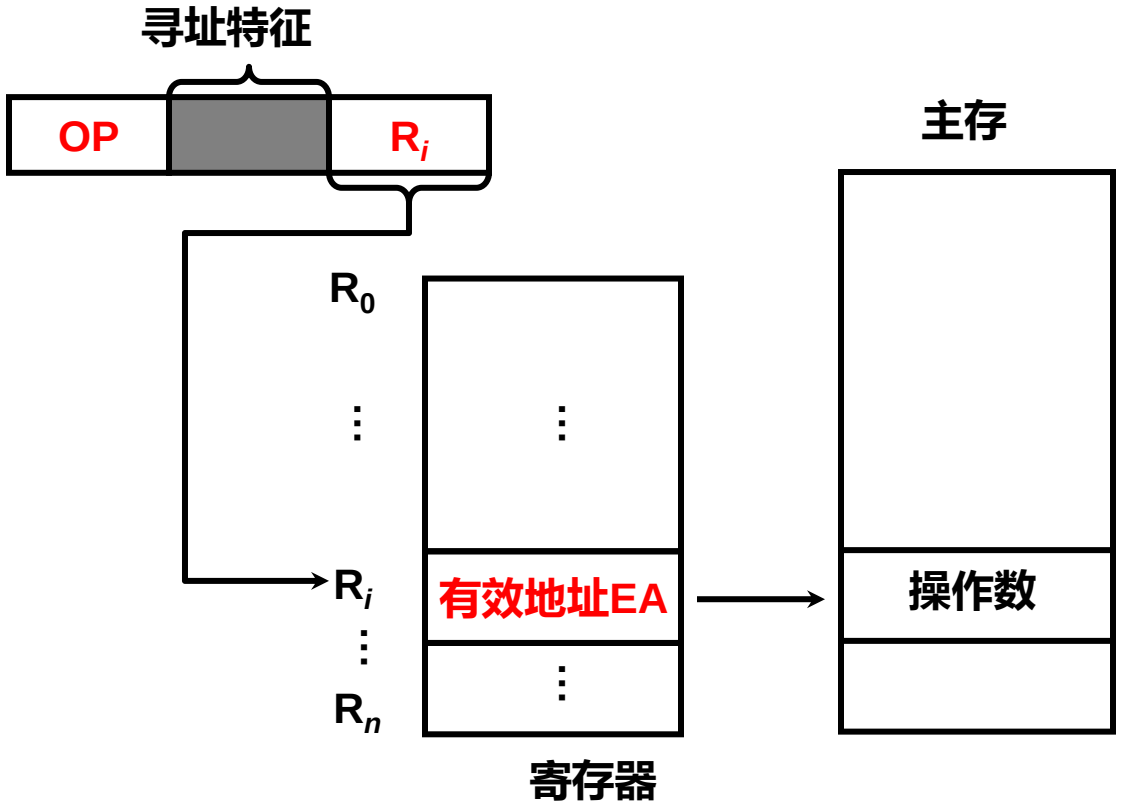
有效地址即为寄存器编号



- 执行阶段不访存，只访问寄存器，执行速度快
- 寄存器个数有限，可缩短指令字长

6. 寄存器间接寻址

EA = (R_i)　　有效地址在寄存器中

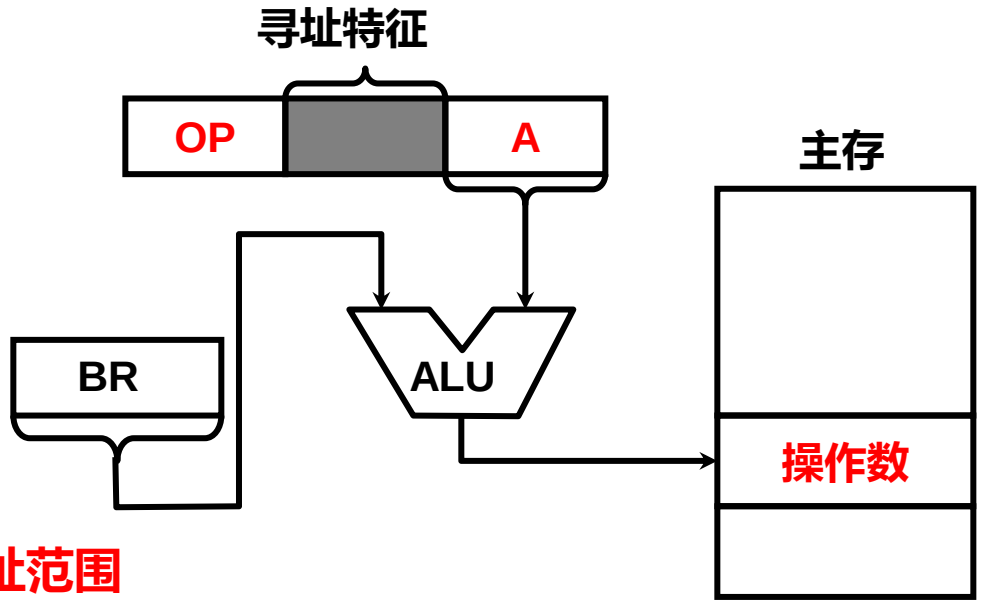


- 有效地址在寄存器中， 操作数在存储器中，执行阶段访存
- 便于编制循环程序

7.3 寻址方式

7. 基址寻址 (1) 采用专用寄存器作基址寄存器

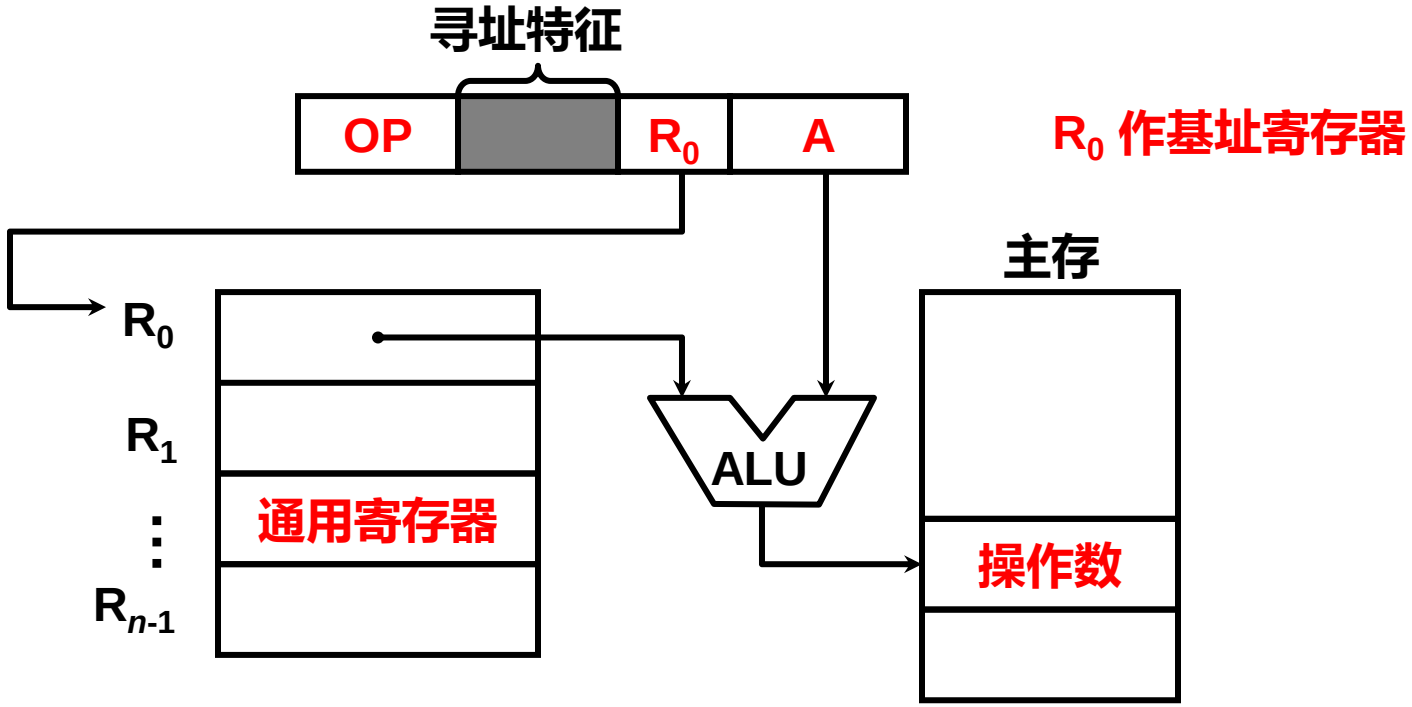
EA = (BR) + A　　BR 为基址寄存器



- 可扩大寻址范围
- 有利于多道程序
- BR 内容由操作系统或管理程序确定
- 在程序的执行过程中 BR 内容不变，形式地址 A 可变

7.3 寻址方式

(2) 采用通用寄存器作基址寄存器



- 由用户指定哪个通用寄存器作为基址寄存器
- 基址寄存器的内容由操作系统确定
- 在程序的执行过程中 R_0 内容不变，形式地址 A 可变

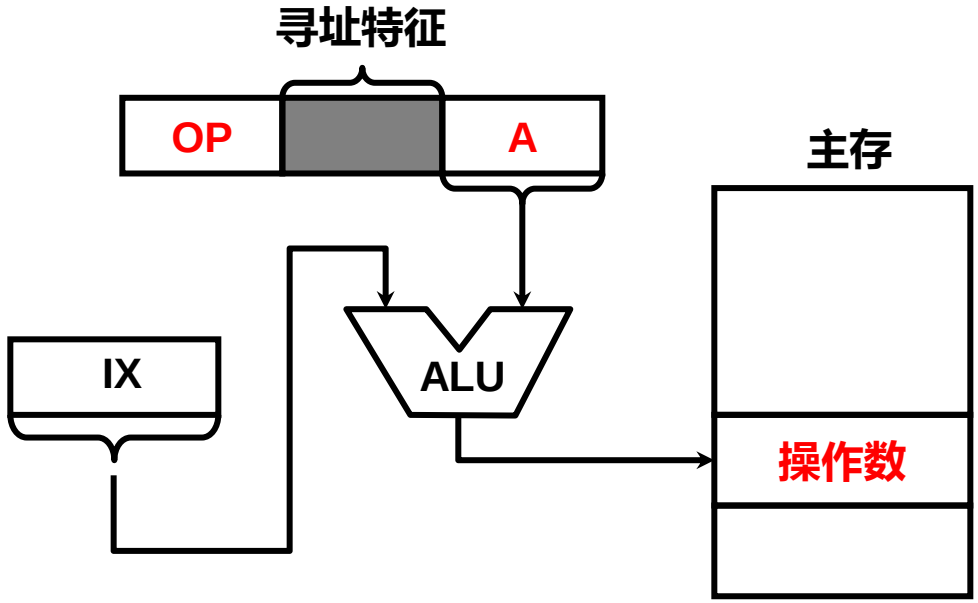
7.3 寻址方式

8. 变址寻址

EA = (IX) +A

IX 为变址寄存器（专用）

通用寄存器也可以作为变址寄存器



- 可扩大寻址范围
- IX 的内容由用户给定
- 在程序的执行过程中 IX 内容可变，形式地址 A 不变
- 便于处理数组问题

变址寻址举例

例7.2　设数据块首地址为 D，求 *N* 个数的平均值

直接寻址

```
LDA    [D]
ADD    [D + 1]
ADD    [D + 2]
  ⋮
ADD    [D + ( N -1 )]
DIV    # N
STA    ANS
```

共 *N* + 2 条指令

变址寻址

```
      LDA    # 0          0 ⟶ ACC
      LDX    # 0          X 为变址寄存器
M     ADD    X, [D]       D 为形式地址
      INX                 (X) +1 ⟶ X
      CPX    # N          (X) 和 #N 比较
      BNE    M            结果不为零则转
      DIV    # N
      STA    ANS
```

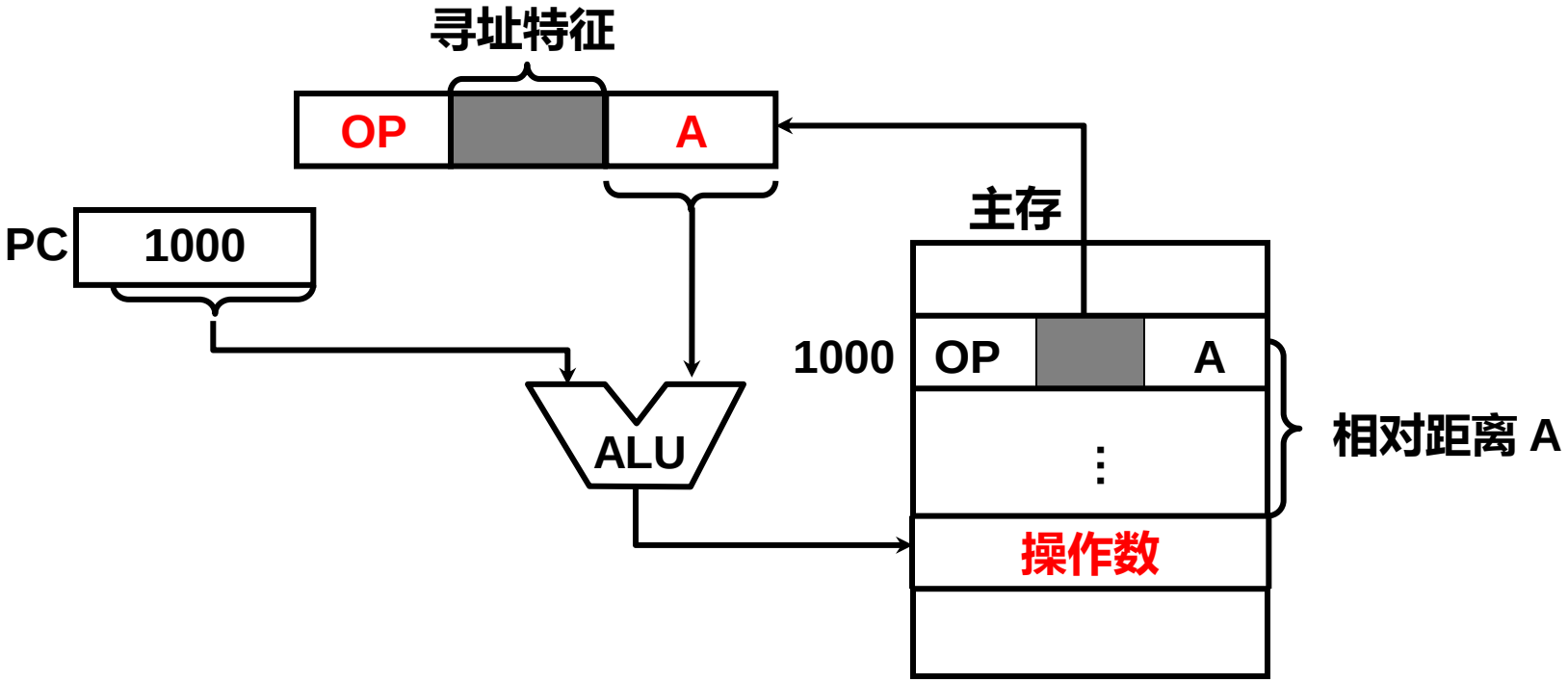
共 8 条指令

7.3 寻址方式

9. 相对寻址

EA = (PC) + A

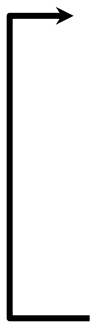
A 是相对于当前指令的位移量（可正可负，补码）



- A 的位数决定操作数的寻址范围
- 程序浮动
- 广泛用于转移指令

7.3 寻址方式

(1) 相对寻址举例



```
LDA    # 0
LDX    # 0
M      ADD    X, D
M+1    INX
M+2    CPX    # N
M+3    BNE    M
DIV    # N
STA    ANS
```

* 相对寻址特征

M+3 BNE M → * – 3

M 随程序所在存储空间的位置不同而不同

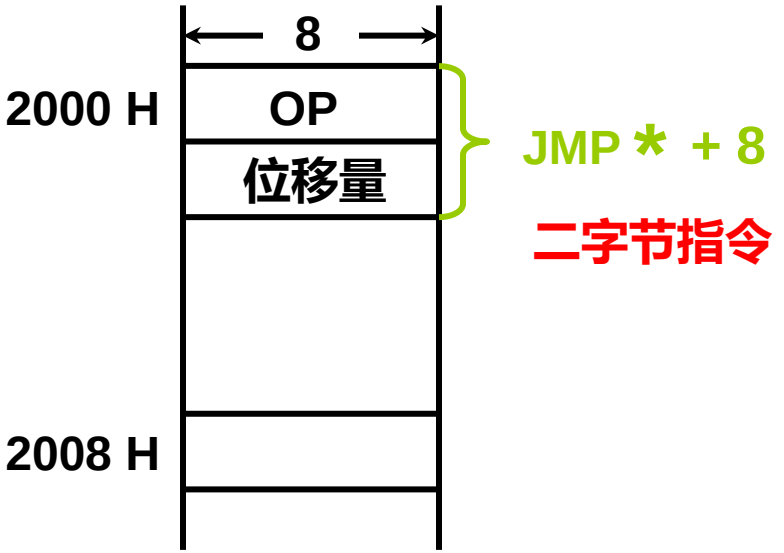
而指令 BNE * – 3 与 指令 ADD X, D 相对位移量不变

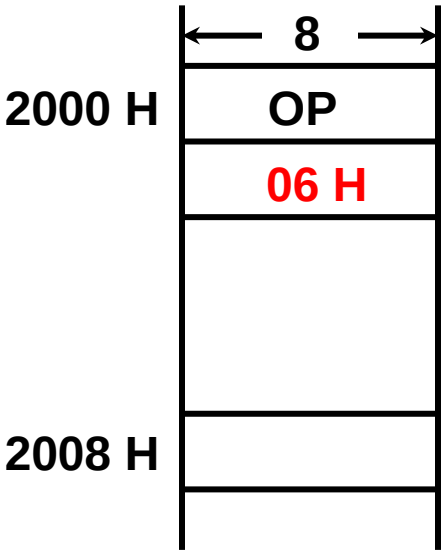
指令 BNE * – 3 操作数的有效地址为

EA = (M+3) – 3 = M

7.3 寻址方式

(2) 按字节寻址的相对寻址举例





设 当前指令地址 PC = 2000H

转移后的目的地址为 2008H

因为 取出 JMP * + 8 后 PC = 2002H

故 JMP * + 8 指令 的第二字节为 2008H - 2002H = 06H

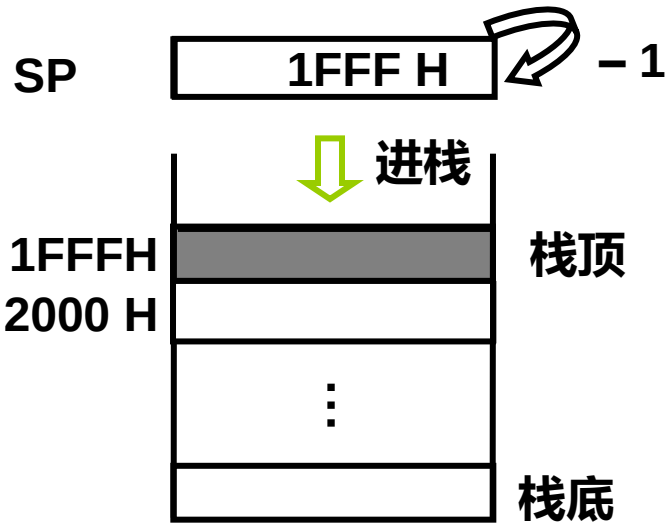
10. 堆栈寻址

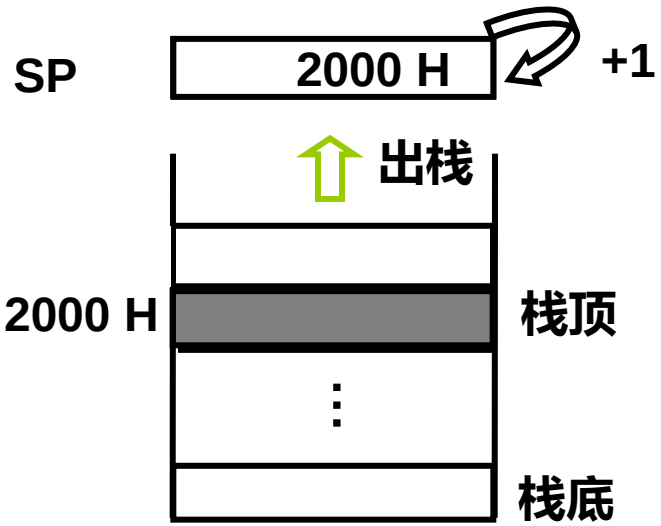
(1) 堆栈的特点

堆栈 { 硬堆栈 — 多个寄存器 — 堆栈计算机
软堆栈 — 指定的存储空间 — 8086计算机

先进后出（一个入出口）　　**栈顶地址** 由 SP 指出

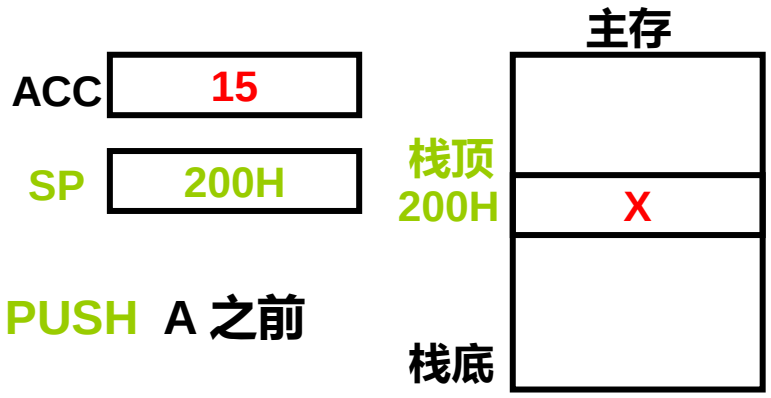
进栈　(SP) – 1 ⟶SP　　**出栈**　(SP) + 1 ⟶ SP

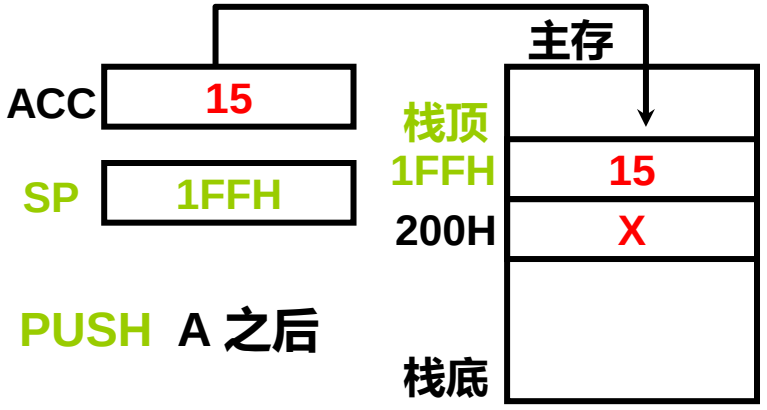


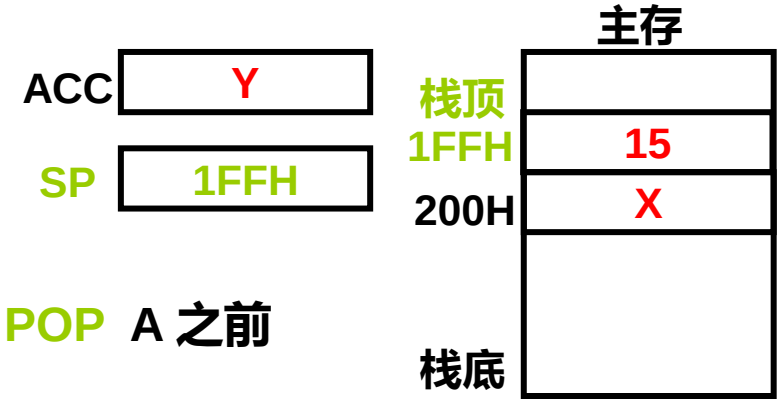


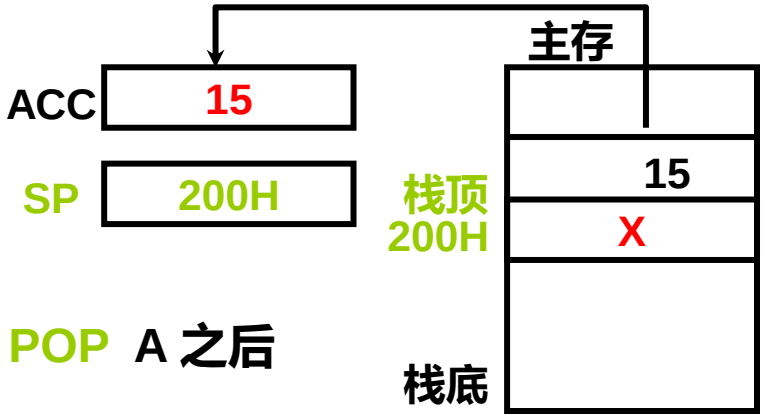
10. 堆栈寻址

(2) 堆栈寻址举例









(3) SP 的修改与主存编址方法有关

① 按 字 编址

进栈　(SP) – 1 ⟶ SP

出栈　(SP) + 1 ⟶ SP

② 按 字节 编址

存储字长 16 位

进栈　(SP) – 2 ⟶ SP

出栈　(SP) + 2 ⟶ SP

存储字长 32 位

进栈　(SP) – 4 ⟶ SP

出栈　(SP) + 4 ⟶ SP

7.3 寻址方式

课后练习：假设（R）=1000，（1000）=2000，（2000）=3000，（3000）=5000，（PC）=4000，则以下寻址方式下访问到的指令操作数的值是多少?

(1) 直接寻址

(2) 间接寻址

(3) 寄存器间接寻址

(4) 相对寻址 -1000

7.4 指令格式举例

一、设计指令格式时应考虑的各种因素

1. 指令系统的 **兼容性**　　（向上兼容）

2. 其他因素

操作类型	包括指令个数及操作的难易程度
数据类型	确定哪些数据类型可参与操作
指令格式	指令字长是否固定
	操作码位数、是否采用扩展操作码技术，
	地址码位数、地址个数、寻址方式类型
寻址方式	指令寻址、操作数寻址
寄存器个数	寄存器的多少直接影响指令的执行时间

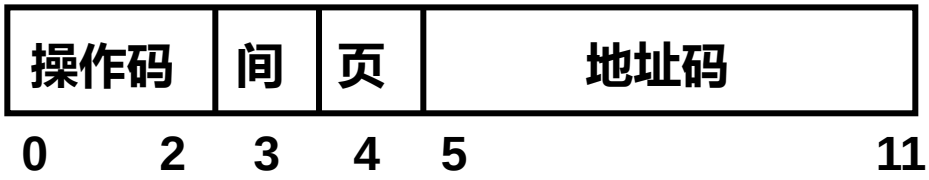
二、指令格式举例

1. PDP – 8

指令字长固定 12 位

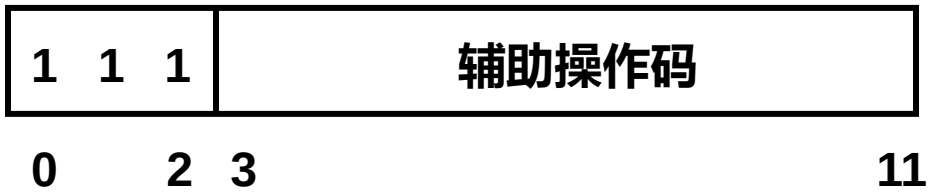
教材P320-321

访存类指令



I/O 类指令

寄存器类指令



采用扩展操作码技术

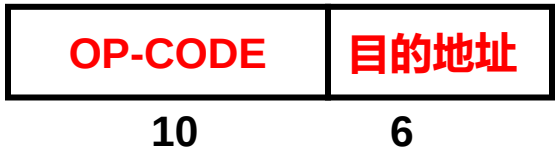
2. PDP – 11

指令字长有 16 位、32 位、48 位三种

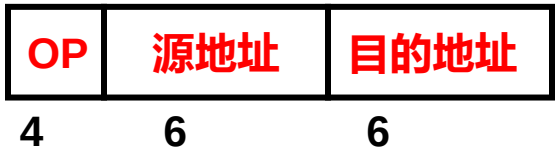
教材P321

零地址 (16 位)

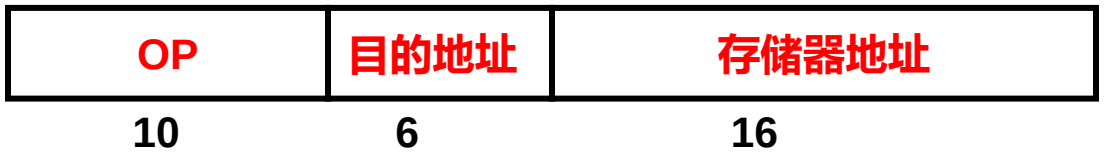
扩展操作码技术



一地址 (16 位)



二地址 R – R (16 位)



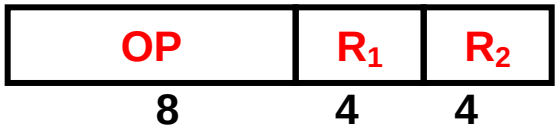
二地址 R – M (32 位)

二地址 M – M (48 位)

3. IBM 360

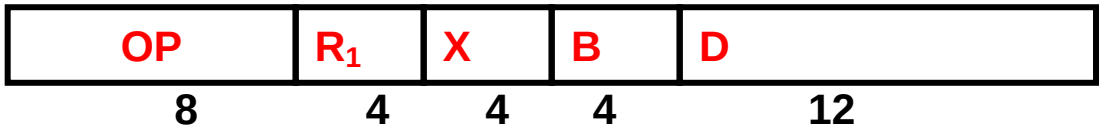
教材P322

RR 格式



二地址 R – R

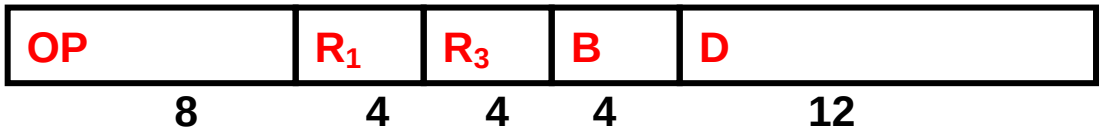
RX 格式



二地址 R – M

基址加变址寻址

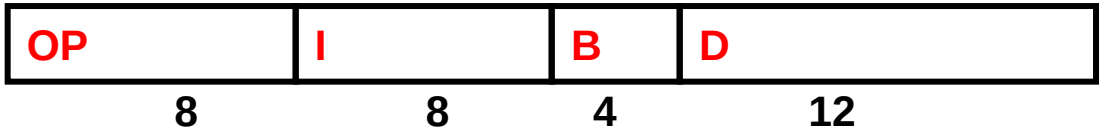
RS 格式



三地址 R – M

基址寻址

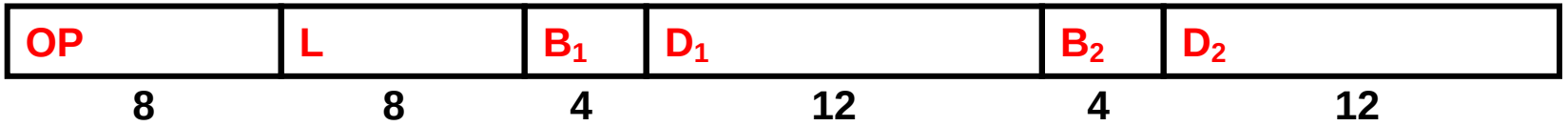
SI 格式



立即数 – M

基址寻址

SS 格式



二地址 M – M

基址寻址

(1) 指令字长　　1 ~ 6 个字节

INC　AX　　1 字节

MOV　WORD PTR[0204], 0138H　　6 字节

(2) 地址格式

零地址　　NOP　　1 字节

一地址　　CALL　　段间调用　　5 字节

CALL　　段内调用　　3 字节

二地址　　ADD AX，BX　　2 字节　　寄存器 – 寄存器

ADD AX，3048H　　3 字节　　寄存器 – 立即数

ADD AX，[3048H]　　4 字节　　寄存器 – 存储器

三、指令格式设计举例-自学*

教材P323-326

课后练习

7.19 CPU 内有 32 个 32 位的通用寄存器，设计一种能容纳 64 种操作的指令系统。假设指令字长等于机器字长，试回答以下问题。

(1) 如果主存可直接或间接寻址，采用寄存器－存储器型指令，能直接寻址的最大存储空间是多少？画出指令格式并说明各字段的含义。

(2) 在满足(1)的前提下，如果采用通用寄存器作基址寄存器，则上述寄存器－存储器型指令的指令格式有何特点？画出指令格式并指出这类指令可访问多大的存储空间？

中国慕课大学
刘宏伟

7.5 RISC 技术

一、RISC 的产生和发展

RISC（Reduced Instruction Set Computer）

CISC（Complex Instruction Set Computer）

80 — 20 规律　　—— RISC技术

- 典型程序中 80% 的语句仅仅使用处理机中 20% 的指令

- 执行频度高的简单指令，因复杂指令的存在，执行速度无法提高

? 能否用 20% 的简单指令组合不常用的80% 的指令功能

二、RISC 的主要特征

- 选用使用频度较高的一些 简单指令，
  复杂指令的功能由简单指令来组合
- 指令 长度固定、指令格式种类少、寻址方式少
- 只有 LOAD / STORE 指令访存，
  其余指令的操作都在寄存器之间进行。
- CPU 中有多个 通用 寄存器
- 采用 流水技术，大部分指令在一个时钟周期内完成
- 采用 组合逻辑 实现控制器
- 采用 优化 的 编译 程序

三、CISC 的主要特征

- 系统指令 **复杂庞大**，各种指令使用频度相差大
- 指令 **长度不固定、指令格式种类多、寻址方式多**
- **访存** 指令 **不受限制**
- CPU 中设有 **专用寄存器**
- 大多数指令需要 **多个时钟周期** 执行完毕
- 采用 **微程序** 控制器
- **难以** 用 **优化编译** 生成高效的目的代码

四、RISC和CISC 的比较 7.5

1. RISC更能 充分利用 VLSI 芯片的面积

2. RISC 更能 提高计算机运算速度

   指令数、指令格式、寻址方式少，
   通用 寄存器多，采用 组合逻辑 ，
   便于实现 指令流水

3. RISC 便于设计，可 降低成本，提高 可靠性

4. RISC 有利于编译程序代码优化

5. RISC 不易 实现 指令系统兼容

锲而舍之，朽木不折；
锲而不舍，金石可镂。

计 算 机 组 成 原 理

复杂指令集计算机CISC

- **CISC的缺陷**
  - **日趋庞大的指令系统不但使计算机的研制周期变长，而且难以保证设计的正确性，难以调试和维护，并且因指令操作复杂而增加机器周期，从而降低了系统性能。**
- **1975年IBM公司开始研究指令系统的合理性问题，John Cocks提出精简指令系统计算机 RISC (Reduce Instruction Set Computer)。**
- 对CISC进行测试，发现一个事实：
  - 在程序中各种指令出现的频率悬殊很大，最常使用的是一些简单指令，这些指令占程序的80%，但只占指令系统的20%。而且在微程序控制的计算机中，占指令总数20%的复杂指令占用了控制存储器容量的80%。
- 1982年美国加州伯克利大学的RISC Ⅰ，斯坦福大学的MIPS，IBM公司的IBM801相继宣告完成，这些机器被称为第一代RISC机。

Top 10 80x86 Instructions

Rank	instruction	Integer Average Percent total executed
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Total	96%

- Simple instructions dominate instruction frequency

（简单指令占主要部分，使用频率高！） back

David Patterson对RISC的回顾

- 计算机发展之初，ROM比起RAM来说更便宜而且更快，所以并不存在片上缓存（cache）这个东西。在那个时候，复杂指令集（CISC）是主流的指令集架构。然而，随着RAM技术的发展，RAM速度越来越快，成本越来越低，因此在处理器上集成指令缓存成为可能。
- 同时，由于当时编译器的技术并不成熟，程序都会直接以机器码或是汇编语言写成，为了减少程序设计师的设计时间，逐渐开发出单一指令，复杂操作的程序码，设计师只需写下简单的指令，再交由CPU去执行。
- 但是后来有人发现，整个指令集中，只有约20%的指令常常会被使用到，约占整个程序的80%；剩余80%的指令，只占整个程序的20%。
- 于是1979年，David Patterson教授提出了RISC的想法，主张硬件应该专心加速常用的指令，较为复杂的指令则利用常用的指令去组合。使用精简指令集（RISC）可以大大简化硬件的设计，从而使流水线设计变得简化，同时也让流水线可以运行更快。
- Patterson教授重申了评估处理器性能的指标，即程序运行时间。程序运行时间由几个因素决定，即程序指令数，平均指令执行周期数（CPI）以及时钟周期。程序指令数由程序代码，编译器以及ISA决定，CPI由ISA以及微架构决定，时钟周期由微架构以及半导体制造工艺决定。对于RISC，程序指令数较多，但是CPI远好于CISC，因此RISC比CISC更快。

指令字长为16位，每个地址码为6位，采用扩展操作码的方式，设计14条二地址指令，100条一地址指令，100条零地址指令。

（1）画出扩展图。（2）给出指令译码逻辑。

（3）计算操作码平均长度。

解：（1）操作码扩展如下：

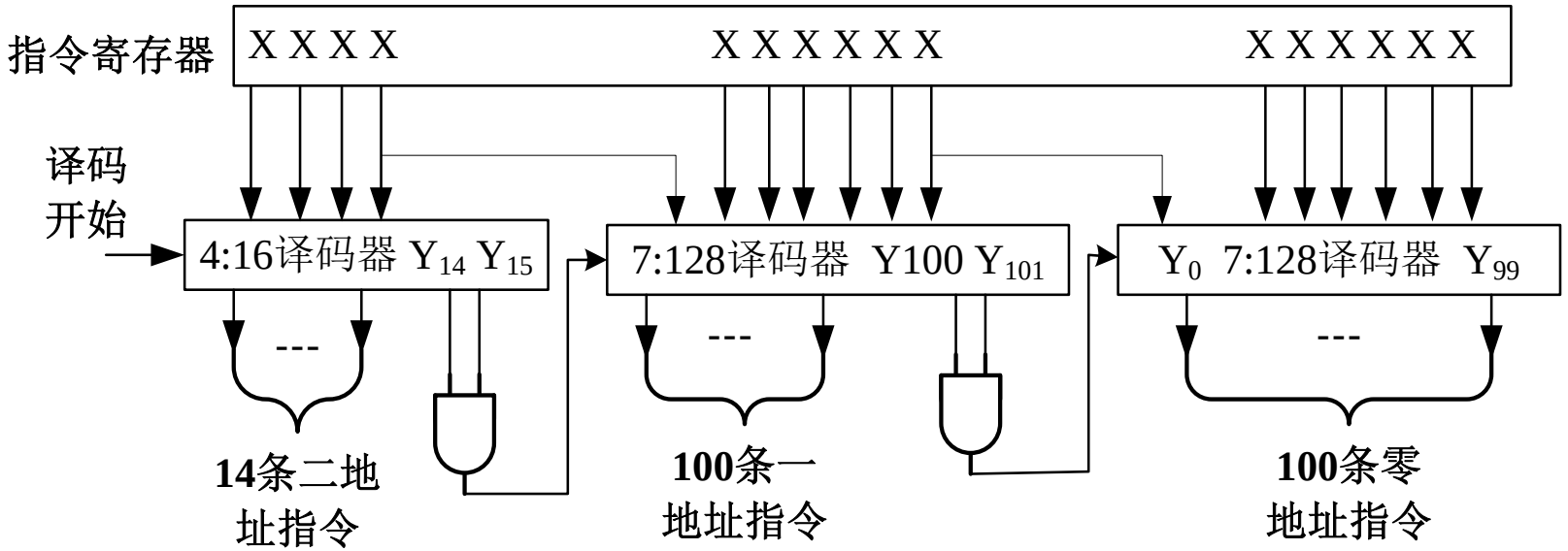
```
0000 xxxxxx  xxxxxx
  ┆      ┆      ┆        14条二地址指令
1101 xxxxxx  xxxxxx
1110 000000  xxxxxx
  ┆      ┆      ┆        100条二地址指令
1111 100011  xxxxxx
1111 100100  000000
  ┆      ┆      ┆        100条二地址指令
1111 100101  100011
```

指令字长为16位，每个地址码为6位，采用扩展操作码的方式，设计14条二地址指令，100条一地址指令，100条零地址指令。

（1）画出扩展图。（2）给出指令译码逻辑。

（3）计算操作码平均长度。

解：（2）指令译码逻辑：



指令字长为16位，每个地址码为6位，采用扩展操作码的方式，设计14条二地址指令，100条一地址指令，100条零地址指令。

（1）画出扩展图。（2）给出指令译码逻辑。

（3）计算操作码平均长度。

解：（3）操作码平均长度

$=（4\times14+10\times100+16\times100）/214\approx12.4$

例：设相对寻址的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，用补码表示。每当CPU从存储器取出一个字节时，即自动完成(PC)+1→PC。

（1）设当前PC值为3000H，问转移后的目标地址范围是多少？

解：（1）由于相对寻址的转移指令为两个字节，第一个字节为操作码，第二个字节为相对位移量，且用补码表示，故其范围为-128～+127，即80H～7FH。又因PC当前值为3000H,且CPU取出该指令后，PC已修改为3002H，因此最终的转移目标地址范围为3081H～2F82H，即3002H + 7FH =3081H至3002H-80H=2F82H

思考：若PC为16位，位移量可正可负，PC相对寻址范围为多大？

解：相对寻址中，PC提供基准地址，位移量提供修改量，位移量为16位可正可负，则相对寻址范围为：（PC）-2^{15}～（PC）$+2^{15}-1$

例：设相对寻址的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，用补码表示。每当CPU从存储器取出一个字节时，即自动完成(PC)+1→PC。

（2）若当前PC值为2000H,要求转移到01BH，则转移指令第二字节的内容是什么？

解：（2）若PC当前值为2000H，取出该指令后PC值为2002H，故转移指令第二字节应为

201BH－002H=19H。

例：设相对寻址的转移指令占两个字节，第一字节是操作码，第二字节是相对位移量，用补码表示。每当CPU从存储器取出一个字节时，即自动完成(PC)+1→PC。

（3）若当前PC值为2000H，指令JMP * -9（*为相对寻址特征）的第二字节的内容是什么？

解：根据汇编语言指令JMP*－9，即要求转移后的目标地址为2000H－09H=1FF7H，但因为CPU取出该指令后PC值已修改为2002H，故转移指令的第二字节的内容应为-11（十进制），写成补码为F5H。